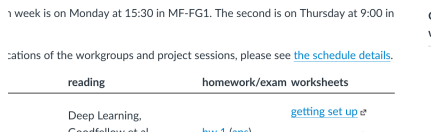
## admin

I you haven't joined a group yet, **do so now**.
Just pick one, and make yourself useful.

Workgroups: times and locations
See CANVAS page **Schedule details**






---

# Probabilistic Models 1: Classifiers

Machine Learning 2019
mlvu.github.io

This lecture will be all about how to use the mechanisms of probability to create a classifier.

## probabilistic classifiers

part 1: **preliminaries**
Probability basics

Information theory
Entropy, Cross-entropy

part 2: **classification**

(Naive) Bayesian classifiers
generative and discriminative classification

Cross-entropy loss
Logistic regression



Young people

One in eight European teenage boys gamble online, says survey

School students across Europe smoke and drink less but there are new public health concerns about excessive screen time

This article is 1 year old

191 145

Alan Travis Home affairs editor

Tuesday 20 September 2016 11.30 BST

The survey found that teenage girls (83%) use social media more regularly than boys, who prefer online gaming. Photograph: ljubaphoto/Getty Images

To start, let's look at the way we use probability *informally*.

Let's say you are a concerned parent, you read this headline and you are shocked by it. You turn to your partner, and you say "that means that the probability that our son is gambling online is 12.5%". Your partner disagrees, you have a good handle on your son's behaviour and his spending. Unless he has a credit card you don't know about, and the probability of that is much lower.

Well, then the probability that Josh, his closest friend, gambles online must be 12.5%. If one in eight teenage boys is gambling, they must be hiding *somewhere*. Your partner disagrees again: probability doesn't enter in to it. Josh is either gambling or he isn't.

Clearly, we need to look at what we mean when we say that a probability of something is such-and-so. There are two commonly accepted ways of looking at it. We'll start with **objective probability**.

*image source: https://www.theguardian.com/society/2016/sep/20/one-in-eight-european-teenage-boys-gamble-online-says-survey*

## objective probability

**frequentism**: probability is only a property of repeated experiments.

In **objective probability**, "the probability that X is the case" represents an objective truth: the probability is the same for everybody.

The most common form of objective probability is **frequentism**. Under the frequentist definition, probability is a property of a (hypothetical) repeated experiment. For instance, take the statement "the probability of rolling 6 with a fair die, is one in six."

The experiment is rolling a die. The outcome we are discussing is the roll resulting in a 6. If we were to repeat the experiment a large number of times, N, then the proportion of times we observe the discussed outcome is close to 1 in 6. More precisely, as N grows, the proportion *converges* to 1 in 6.

Saying "the probability is one in six", is equivalent to saying "if I throw the die repeatedly, the relative frequency of sixes will converge to 1 in 6 as the number of repeats grows."

---

Young people

# One in eight European teenage boys gamble online, says survey

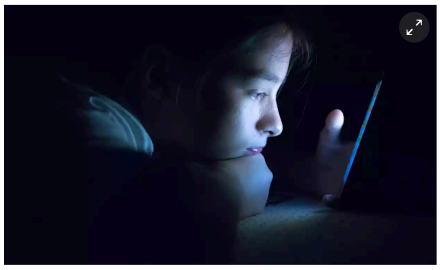
School students across Europe smoke and drink less but there are new public health concerns about excessive screen time

This article is 1 year old

< 191 145

Alan Travis Home affairs editor

Tuesday 20 September 2016 11.30 BST



The survey found that teenage girls (83%) use social media more regularly than boys, who prefer online gaming. Photograph: ljubaphoto/Getty Images

Under objective probability the statement "the probability that Josh is gambling is 12.5%" is indeed nonsense. There is no experiment we can imagine where Josh "turns out" to be a gambler one in 8 times. He either gambles or he doesn't.

What we *can* say is that the probability that a teenage boy drawn randomly from the European population gambles online is 12.5%. This is an experiment we can repeat, and at every repetition, we choose a different boy, so we get a different outcome.

We should also note that our statement is not *precisely* correct. The actual probability is a number we don't know. This is what happens in practice: the probability of X happening is p. We don't know p, but we do know that there is some experiment for which the proportion of successful trials (X happens) converges to p with repeated trials. We repeat a large number of trials, check the proportion of times p happened, and use that as an estimate of the true p. That is also what happened in the research behind this article. We don't know precisely how many teenage boys gamble online, so the researchers found a way to estimate the total proportion

## subjective probability

**Bayesianism:** probability is an expression of our **uncertainty** and of our beliefs.

7

The alternative to objectivism is **subjectivism**. It states that probability expresses our uncertainty. If X is a boolean variable, one that is true or not true, and we are uncertain whether X is true, we can assign a probability to X being true. A probability of 0.5 means we are entirely ambivalent, a probability of 0.75 means we think X is pretty likely, and a probability of 1 means we're entirely sure.

**Bayesianism** is the main form of subjective probability. It builds on Bayes' rule (more on that later) to tell us how we should use observations to update our beliefs.

---

Young people

One in eight European teenage boys gamble online, says survey

School students across Europe smoke and drink less but there are new public health concerns about excessive screen time

This article is 1 year old

< 191 145

Alan Travis Home affairs editor

Tuesday 20 September 2016 11.30 BST

The survey found that teenage girls (83%) use social media more regularly than boys, who prefer online gaming. Photograph: ljubaphoto/Getty Images

Under subjectivism, we *can* say "the probability that our son is gambling is 12.5%" We don't know what he gets up to, so even though there is a definite objective answer, *we* are uncertain. If we know only this headline, we may well pick 12.5% as our belief that our son is gambling. Of course, as noted before we have a lot more knowledge about our son. We know he goes to bed on time, we know where gets his money, he probably doesn't have a secret credit card. So even though the probability for a random teenage boy would be 12.5%, the probability for our son is actually much lower, because we have extra information.

This is the fundamental difference between the two views: under *frequentism*, probability is defined as an objective property of the world. The probability of X is the same for all people regardless of what we know or don't know. Under *Bayesianism*, probability is an expression of a subjective property: it can change from one person to the next, and if we learn new information, it can change from one moment to the next. If we find out that our son *does* have a secret credit card, the probability that he is gambler, suddenly jumps dramatically

Note that Bayesianism, in a sense encompasses frequentism. If we know the mechanics of an experiment, our belief about the outcome coincides with the frequentist use of the relative frequency. Bayenism just extends the definition to allow for personal beliefs that are not objectively true.

## subjectivism vs. objectivism

A *disambiguation* of the word probability.

Leads to fundamentally different ways of doing statistics.

Is machine learning a probabilistic discipline?

If so, is it subjective or objective?



Note that you don't have to commit to one view or another. At heart subjective and objective probability are just ways to be more precise about what the word probability actually means. You can use the subjective definition one day and the objective definition the next (especially in informal settings).

However, once you start doing statistics, the two definitions lead to fundamentally different approaches (which we'll see in more detail later).

Since machine learning is often seen as a dance form of statistics, you may ask whether it is usually seen as using subjective or objective probability. I can't give you a commonly accepted answer, I think opinions differ.

My view is that Machine Learning, while being statistical in nature, is not fundamentally probabilistic. The fundamental principles of machine learning can be defined and explained without recourse to probability theory (and indeed, we have done so for most of the start of the course). The fundamental goal of (offline) machine learning is to minimise test set loss given only a training set, and some hint as to the relation between the two datasets.

Of course, even if machine learning is not fundamentally probabilistic, probability has proven to be a very powerful tool (much like linear algebra and calculus), in helping us solve this problem.

## probability theory

Basic ingredients

- sample space
- event space
- probability function p(…)
- random variable



The mathematical definition of probability, studied in the field of probability *theory*, is entirely distinct from the question of what the definition of probability is. Both frequentists and Bayesians use the same basic framework.

We'll go through the ingredients quickly.

## sample space

$\Omega = \{\text{heads}, \text{tails}\}$

$\Omega = \{1, 2, 3, 4, 5, 6\}$ <- discrete sample spaces

$\Omega = \{(1, 1), (1, 2), \ldots, (6, 6)\}$

$\Omega = \mathbb{R}$ <- continuous sample space



First the **sample space**. These are the single outcomes or truths that we wish to model. If we flip a coin, our sample space is the set of the two outcomes heads and tails.

We can have discrete sample spaces or continuous ones. A discrete sample space can also be infinite: consider flipping a coin and counting how many flips it takes to see tails. In this case any number of flips is possible, so the sample space is the natural numbers (although any number larger than 20 will get an astronomically small probability).

## event space

$E = \{\{\}, \{1\}, \{2\}, \{3\}, \ldots, \{1, 2\}, \ldots, \{1, 2, 3, 4, 5, 6\}\}$

Events are the things that have probability: subsets of the sample space. All even throws, all throws higher than three, etc.

powerset: the set of all subsets
sigma-algebra: for continuous sample spaces.



For continuous spaces, not everything can be an event. We need to make sure that our event space is a "sigma algebra." For our purposes, we don't need to worry about this.

## random variable

A way to describe events

D "takes values" 1, 2, 3, 4, 5, 6

p(D = 4), p(D > 3), p(D is even) etc

Random variables in ML:

- features i of instance j: $X_i^j$
- class of instance j: $Y^j$
- Model (parameters): M



Random variables have a confusing and convoluted definition, so we'll just give you the intuitive interpretation.

Random variables help us to describe events.

## shorthand

p(X = 0): the probability that X takes the value 0

**A number between 0 and 1**

p(X = x): the probability that X takes the value x.

**A function of x.**

$$p(X = x) = \begin{cases} \frac{1}{4} & \text{if } x = 0 \\ \frac{3}{4} & \text{if } x = 1 \end{cases}$$

p(X), p(x): shorthand for P(X = x)



After all this, statements about probability can get quite convoluted. Here are some common shorthands you'll probably encounter.

Since we usually know which outcomes below to which random variables, p(X) and p(x) can both be used as shorthand for p(X=x). Note that in these cases, x stands for some specific value, and X stands for the random variable.

## probabilities and concepts

for random variables X and Y

joint probability: p(X, Y)

marginal probability: p(X)

conditional probability: p(X | Y)

(conditional) independence

Bayes' theorem

Now that we have the basic language of probability theory in place, we can look at some of the important concepts. We will quickly review these five concepts.

Note that we have a single event/sample space, and the random variables X and Y will help us describe the events that we're interested in.

## running example

Age = {young, teen, old}

Teeth = {healthy, unhealthy, fake}



We will use the following running example: we sample a random person from the Dutch population and we check their age and the health of their teeth (binning the results into three categories for each variable).

We want to ask questions like:

- what is the probability of seeing an old person?
- what is the probability that a young person has fake teeth?
- does a person's age influence the health of their teeth, or is there no relation?

## joint probability

p(Age = old & Teeth = healthy)

p(Age, Teeth):

| | | T | | |
|---|---|---|---|---|
| | | h | u | f |
| A | y | 5/18 | 3/18 | 1/18 |
| | t | 1/18 | 1/18 | 2/18 |
| | o | 1/18 | 1/18 | 3/18 |

17

The joint distribution is the most important distribution. It tells us the probability of each **atomic event**. That is, if we specify a single value for each random variable, we get a probability in return.

Since we have two random variables in our example, we can specify the joint distribution in a small table. Note that each cell in this table corresponds to one element of our sample space. The probabilities of these events sum to one.

Note that p(Age = old & Teeth = healthy) refers to a single value (1/18), because we have specified the event. p(Age, Teeth) does not refer to a single value, because the variables are not instantiated, it represents *a function of two variables* (i.e. the whole table).

---

## marginal probability

| | | T | | | |
|---|---|---|---|---|---|
| | | h | u | f | |
| A | y | 5/18 | 3/18 | 1/18 | 9/18 |
| | t | 1/18 | 1/18 | 2/18 | 4/18 |
| | o | 1/18 | 1/18 | 3/18 | **5/18** |
| | | 7/18 | 5/18 | 6/18 | |

p(Age=old)

18

If we want to focus on just one random variable, all we need to do is sum over the rows or columns. For instance, the probability that Age=old, regardless of the value of Teeth, is the probability of the event {(o,h), (o,u), (o,f)}. Because we can write these sums in the *margins* of our joint probability table, this process of "getting rid" of a variable is also called **maginalizing out** (as in "we marginalize out the variable Teeth"). The resulting distribution over the remaining variable(s) is called a **marginal distribution**.

## marginal probability

$$p(y) = p(y, h) + p(y, u) + p(y, f)$$

in general, for joint distribution p(X, Y):

$$p(X=x) = \sum_{y \text{ in } Y} p(x, y)$$



## conditional probability

$$p(T = f \mid A = y) = p(f, y) / p(y) = 1/9$$

| | | T | | |
|---|---|---|---|---|
| | | h | u | f |
| | y | 5/18 | 3/18 | **1**/18 |
| A | t | 1/18 | 1/18 | 2/18 |
| | o | 1/18 | 1/18 | 3/18 |



The conditional probability is the probability over one variable, if the value of another is known.

Here, we see the probability that a person has false teeth, given that they're young.

The conditional probability p(X=x|Y=y) is computed taking the joint probability of (x, y) and normalising by the sum of the probabilities in the row or column corresponding to the part that's given in the conditional.

Imagine we're throwing darts at this table, and the probability of hitting a certain cell is the joint probability indicates in the cell. The conditional probability p(T=f|A=y) is the probability that the dart hits the (y, f) cell, given that it's hit the y row.

## conditional probability

$$p(X = x|Y = y) = \frac{p(X = x, Y = y)}{\sum_{x'} p(X = x', Y = y)}$$

$$= \frac{p(x, y)}{p(y)}$$



Note that the denominator is just the marginal probability

## useful

$$p(x, y) = p(x \mid y)p(y)$$



The conditional probability can be rewritten like this, showing a kind of decomposition of the joint probability. This comes up a lot, so it's useful to make a mental note of it.

## continuous

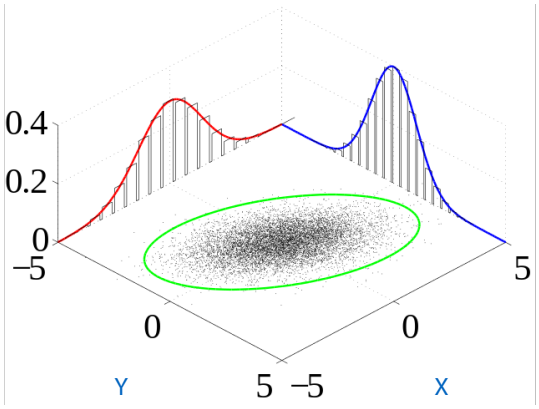



image source: By Ikamusume Fan - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=30432580



Here is what these concepts look like with *continuous* random variables (a bivariate normal distribution in this case). The joint probability distribution is represented by the point cloud in the middle. Marginalizing out either variable results in a univariate normal (the red and blue distributions).

The conditional distribution corresponds to a vertical or horizontal slice through the joint distribution (and also results in a univariate normal.

## independence

X and Y are independent if

$p(X, Y) = p(X)p(Y)$

which implies $p(X|Y) = p(X)$

X and Y are conditionally independent given Z if

$p(X, Y \mid Z) = p(X \mid Z)p(Y \mid Z)$



If two variables X and Y are independent, then knowing Y will not change what we know about X.

Conditional indolence means that the two variables are dependent, but their dependence is entirely explained by a third variable Z. If we condition on Z, the variables become dependent.

For an example: consider two people a and b who work in different cities in the Netherlands. Define random variables A and B describing whether or not a and b respectively are late for dinner. They live far enough away, that the two events are entirely unrelated, except that when it snows in the Netherlands everything shuts down. Represent the event of snow by the random variable S. Now, if I know that A was late for dinner, there is a small probability that that was caused by snow. This the probability that B was late for dinner as well slightly increases. However, if I know that it didn't snow (I condition on S), knowing that A was late for dinner doesn't influence the probability of B being late for dinner at all.

## conditional independence

A: Alice is home in time for dinner
B: Bob is home in time for dinner
G: a monster attacks the city

$p(A, B|G) = p(A|B)\ p(B|G)$

$p(A|G, B) = p(A|G)$

Conditional independence comes up a lot, and it can be tricky to wrap your head around at first, so here's an example.

Imagine two people who work in different areas of a very big city. In principle, they work so far apart that whether or not they arrive home in time for dinner is completely independent. Knowing whether or not Alice is late for dinner tells you nothing about whether Bob is home in time for dinner. No aspect of their lives (weather, traffic) intersect in a meaningful way, except one.

Very rarely, a large monster attacks the city. In that case, all traffic shuts down and everybody is late for dinner. That means that if we know that Bob is late for dinner, there is a slight chance that it's because of the monster, which should slightly raise the possibility that Alice is late for dinner. However, once we know whether or not the monster has attacked, knowing that Bob is late provides no additional information.

---

## Bayes' rule

**the inversion problem:**

It's easy to express the probability of an observable given some hidden cause (assuming we have a model of the world). However, we usually want the opposite



In short, we need a way to "turn around" the conditional probability. If we know $p(X|Y)$, how do we work out $p(Y|X)$?

$$p(Y \mid X) = \frac{p(X \mid Y)\, p(Y)}{p(X)}$$



## conditional probability and Bayes' rule

$$p(X \mid Y) = \frac{p(Y, X)}{p(Y)}$$

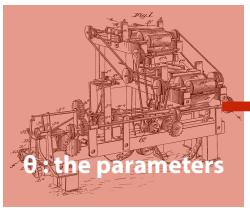
definition of conditional probability

$$= \frac{p(Y \mid X)p(X)}{p(Y)}$$

see **slide 22**

## learning

"machine"




Observed data

We understand the machine, p(Data | θ) is known. But we observe only the Data (and the input) and we want to know θ.

29

Here is an analogy for the way probability is usually applied in statistics and machine learning. We assume some "machine" (which could be any process, the universe, or an actual machine) has *generated* our data, by a process that is partly deterministic and partly random. The configuration of this machine is determined by its **parameters** (theta). Theta could be a single number, several numbers or a even a complicated data structure.

We know how the machine works, so if we know theta, we know the probability of each dataset. The problem is that we only observe the data, and we cannot look inside the machine.

## "frequentist" learning

**Maximum likelihood** estimation

$$\hat{\theta} = \arg\max_{\theta} p(X \mid \theta)$$

The function L(θ) = p(X|θ) is called the *likelihood*.

30

The frequentist approach is to decide some reasonable criterion to select a particular parameter value based on the data we've observed.

One of the most common criteria is that we should prefer the theta for which the probability of seeing the data that we saw is highest. This is called the *maximum likelihood principle*.

Note that the likelihood is a function of the parameters, not the data.

## fitting a normal distribution

observations : $X^1, X^2, \ldots$

$$\hat{\mu}, \hat{\sigma} = \arg\max_{\mu,\sigma} p(X^1, X^2, \ldots \mid \mu, \sigma)$$

$$= \arg\max_{\mu,\sigma} \prod_i N(X^i \mid \mu, \sigma)$$

31

Imagine that we see some data (a sequence of numbers), and we assume that each item in the sequence was drawn independently from the same one-dimensional normal distribution. We commonly model this as a sequence of random variables, which we assume to be **independent and identically distributed (iid)**. That is, they were all drawn from the same distribution and each draw is independent from the others.

The maximum likelihood estimates for the parameters mu and sigma of this distribution are those values that maximize the probability of the data. In this case, we've assumed that each

---

## Bayesian learning

$$p(\theta \mid X)$$

32

Choosing a single best value for the parameter(s) is called a **point estimate**.

Bayesians do not care for point estimates. Since probability expresses a belief, we can just express our belief about the state of the machine theta as another probability distribution. That way we don't have to guess the value of the parameters, we know for every single possibility how likely it is.

Note that X here is a shorthand for any kind of data: it could be a sequence of numbers, or a full matrix of features and instances.

$$p(\theta \mid X) = \frac{p(X \mid \theta)p(\theta)}{p(X)}$$

$p(\theta)$ the **prior**

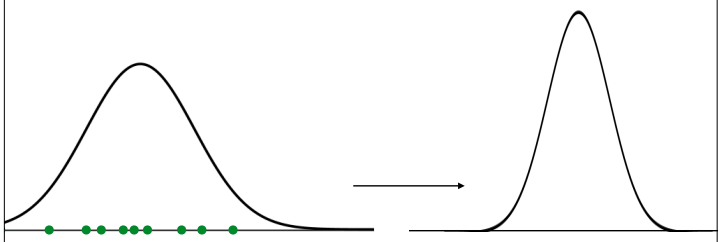
$p(\theta \mid X)$ the **posterior**

33

The **prior** expresses your belief in the values of the parameters before seeing the data, the **posterior** tells you your beliefs afterwards.

The prior is the only part that cannot be computed. You have to choose it based on your actual beliefs (more often, it's chosen with a specific form to ensure that the math works out in a particular way).

## a Bayesian fits a normal distribution



$p(\mu \mid \mathbf{X})$

34

How do you "produce" a model as the output of a statistical process? Either state the model of the distribution, and work out what the parameters are (here, the distribution over mu is normal, and we can work out what the mean and standard deviation should be given the data, and a prior over the parameters). This usually requires that the prior has a very specific form.

If this can't be done, we can also leave the exact definition of the resulting distribution unclear, and try to *sample* from it instead. In that case, learning a good model reduces to sampling from the posterior. Often this looks a lot like iterative search for a good model, like simulated annealing.

## Maximum a Posteriori (MAP) criterion

$$\hat{\theta} = \arg\max_{\theta} p(X \mid \theta) p(\theta)$$

35

Finally , the **MAP criterion** is like the maximum likelihood model, but with a prior. Like a good compromise, it leaves everybody unhappy.

Pure frequentists disown it, because it expresses a probability over an objective value (the parameter) and pure Bayesians disown it, because it's a point estimate, when you can also have a distribution.

Nevertheless, it's a very useful concept, and you will see it mentioned a lot in ML literature.

Note that in Machine Learning research, people rarely care about being pure Bayesians or pure frequentists. They tend to use what works and mix and match techniques from both camps.

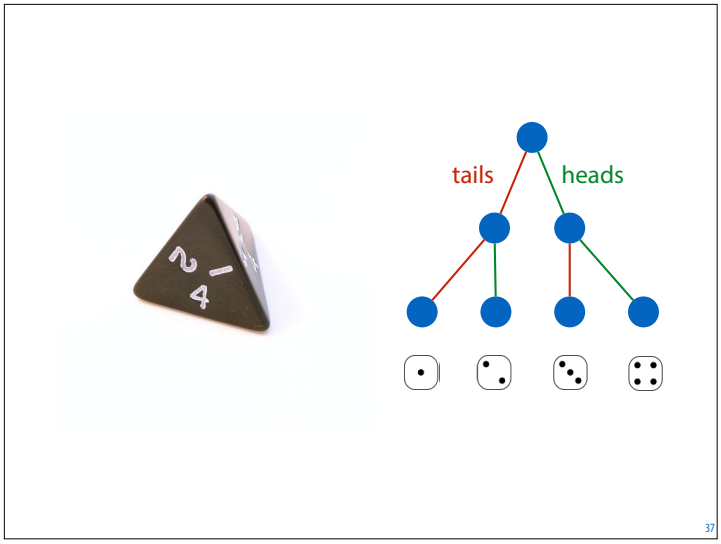
---

## information theory


36

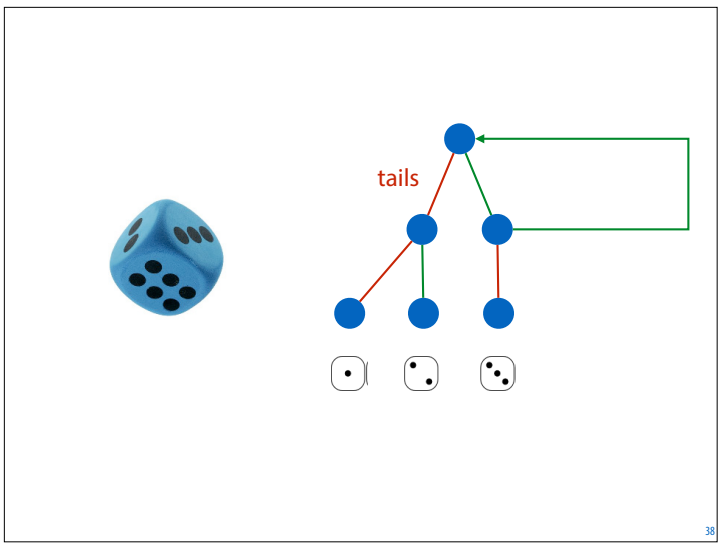
Information theory is all about the relation between encoding information and probability theory.

Imagine you're on holiday, and you've brought your travel monopoly. Unfortunately, the dice have gone missing. You do however, have a coin with you. Can you use the coin flip to simulate the throw of a six sided  die?




For a four sided die, the solution is easy. We flip the coin twice, and assign a number to each possible outcome.

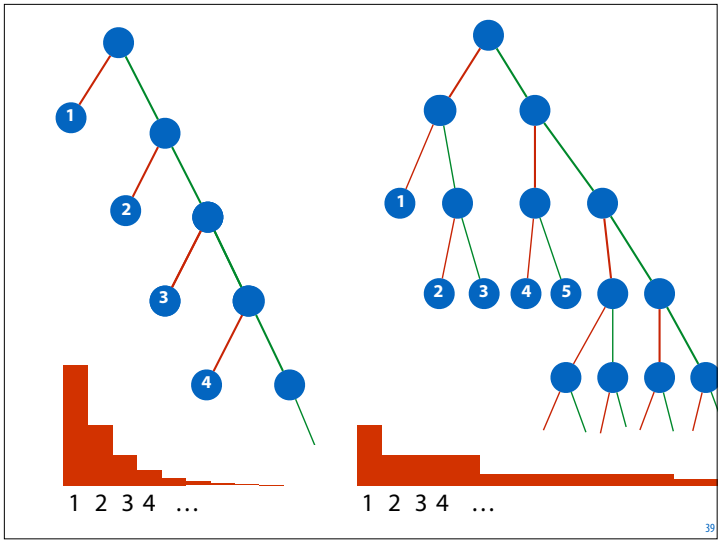
source: **http://www.midlamminiatures.co.uk/blackpolydice/D4Black.html**



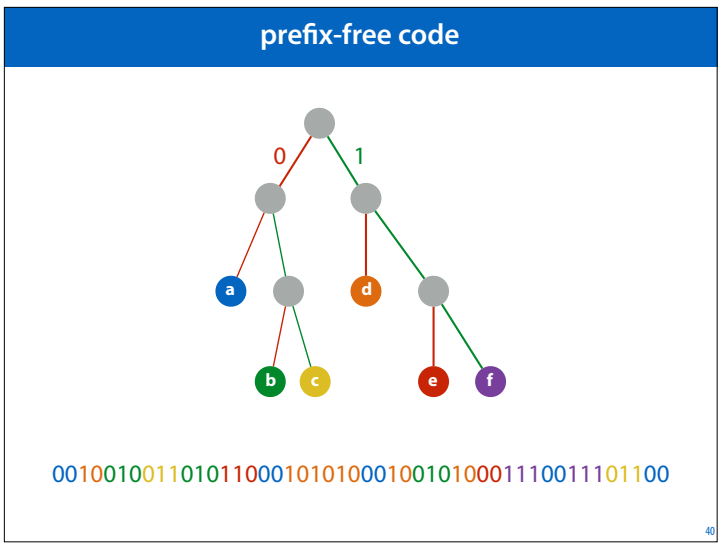

A six sided die is more tricky. We'll show the solution for three "sides" (you can just add another coin flip to decide whether it'll be 1,2,3 or 4,5,6.)

The trick is to assign the fourth outcome to a "reset". If you throw two heads in a row, you just start again. Theoretically you could be coin flipping forever, but the probability of resetting more than five times is already less than one in one-thousand.

For now let's stick with trees where each outcome is represented by only one leaf (and accept that the six-sides die cannot be perfectly modelled with a coin). What distributions can we model with a coin in this way, if we require each outcome to be represented by one leaf in the tree?

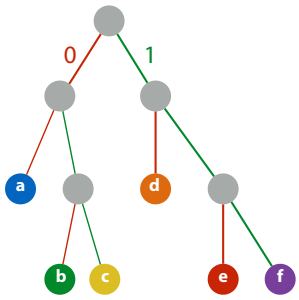



Here are two examples: an exponentially decaying distribution, and a (roughly) polynomially decaying one.




These kinds of trees are called prefix-free trees, because they assign a *prefix free code* to the set of outcomes (we just replace heads and tails with zeros and ones). The benefit is that if we want to encode a sequence of these outcomes, we can just stick the code one after another and we won't need any delimiters. A decoder will know exactly where each codeword ends and the next begins.

## codelengths and probabilities




L(x): length of code for x

$$p(x) = \frac{1}{2} \times \ldots \times \frac{1}{2}$$

$$= \left(\frac{1}{2}\right)^{L(x)}$$

$$= 2^{-L(x)}$$

$$L(x) = -\log_2 p(x)$$

41

Every prefix tree defines a probability distribution and a code. What about the other way around? Can we find a tree for any given probability distribution?

We already saw that some distributions (like a six-sided die) cannot be represented exactly. But how close can we get?

## arithmetic coding

There exists an algorithm which provides for any p(x), a prefix-free code such that

$$|-\log_2 p(x) - L(x)| \leqslant 1$$

Thus, if we ignore this minor inaccuracy, or if we allow L(x) to take non-integer values, we may

equate codes with probability distributions.

42

It turns out we can model any distribution in such a way that the biggest difference in codelength is no larger than a bit.

If we handwave this difference, we can equate codes with probability distributions: every code gives us a distribution and every distribution gives us a code. The higher the probability of an outcome, the shorter its codelength.

## entropy

p(X=x): data source

If we encode X with the ideal code for p, what is our expected codelength?

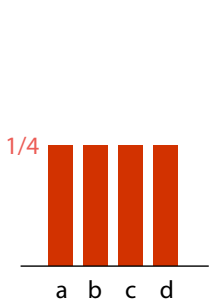
$$H(p) = \mathbb{E}_p L(x)$$

$$= \sum_{x \in X} p(x) L(x)$$

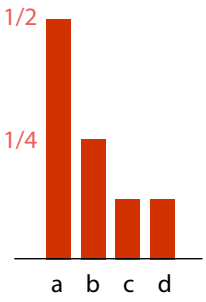
$$= -\sum_{x \in X} p(x) \log p(x)$$

43

The entropy of a distribution is the expected codelength of an element sampled from that distribution.




H(p) = 2 bits




H(p) = 1.75 bits

44

The more uniform our distribution is (the more unsure we are) the higher the entropy.

On the right, we know something about our distribution, for instance that a is very likely, so we can make the codeword for a a little shorter, reducing the expected codelength (the entropy). On the left, we have no such options, so the entropy is maximal (equal to $\log_2 N$).

## cross entropy

p(X): source of our data
q(X): our model

Cross entropy: expected codelength if we use q, but the data comes from p.

$$\begin{aligned} H(p, q) &= \mathbb{E}_p L^q(x) \\ &= -\sum_{x \in X} p(x) \log q(x) \end{aligned}$$



What if we don't use the code that corresponds to the source of our data p to encode our data, but some other code based on distribution q. What is our expected codelength then? This is called the *cross entropy*.

The cross entropy is minimal when p=q (and equal to the entropy). We can conclude two things:

- The code corresponding to p provides the best expected codelength.
- The cross entropy is a good way to **quantify the distance between two distributions** (because it's minimal when the two are the same).

## Kulback-Leibler divergence

Expected difference in codelength between p and q.

Or, difference in expected codelength.

$$\begin{aligned} KL(p, q) &= H(p, q) - H(p) \\ &= -\sum_{x \in X} p(x) \log \frac{q(x)}{p(x)} \end{aligned}$$



The cross entropy is a nice measure, but it's not zero when p and q are equal. Instead, it's equal to the entropy of p.

To get a measure that is zero when the two are equal, we can just subtract the the entropy of p. This is called the Kulback-Leibler (KL) divergence.

## summary: information theory

We can equate code and distributions. The distribution that assigns x a high probability, equates to a code that assigns x a low codelength.

Specifically L(x) = -log p(x)

H(p):

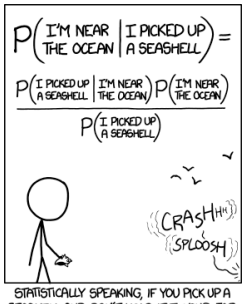
Good measure of *uniformity* of p.

H(p, q) and KL(p, q):

Good measures of *distance* between model and truth.



## break




STATISTICALLY SPEAKING, IF YOU PICK UP A SEASHELL AND *DON'T* HOLD IT TO YOUR EAR, YOU CAN PROBABLY HEAR THE OCEAN.



source: https://xkcd.com/1236/

## classification

$X = X_1, X_2, X_3, \ldots$: random variable for instance.

Y: random variable for class {pos, neg}

$P(Y=pos \mid X) = 0.1 \qquad P(Y=neg \mid X) = 0.9$



For the second half of this lecture we will focus on building **probabilistic classifiers**. These are classifiers that return not just a class for a given x (or a ranking) but a probability over all classes.

This can be very useful. We can use the probabilities to extract a ranking (and plot an ROC curve) or we can use the probabilities to assess how certain the classifier is. If we don't want the probabilities, we can just turn it into a regular classifier by picking the class with the highest probability.

Note that a probabilistic classifier is also immediately a ranking classifier and a regular classifier.

## two approaches

**discriminative classifier:**
learn a function for p(Y|X) directly

**generative classifier:**
$p(Y|X) \propto p(X|Y)p(Y)$



A **discriminative classifier** learns p(Y|X) directly. It functions as a kind of regression, mapping x to a vector of class probabilities.

A generative classifier instead focuses on learning a distribution on the feature space, conditioned on the class. This distribution is then combined with Bases' rule to get the probability over the classes, conditioned on the data.

## generative classifiers

**Bayes optimal classifier**
Provably optimal (given certain assumptions). Usually too expensive to compute.

**Bayes classifier**
Reasonable approach for low dimensional data.

**Naive Bayes classifier**
Simple, cheap and effective for high-dimensional.



Here are three approaches, arranged from impractical but entirely correct to highly practical, but based on largely incorrect assumptions.

We won't discuss the Bayes optimal classifier today, but it's worth knowing that it exists, and that it means something different than a (naive) Bayes classifier.

## Bayes classifier

$$p(Y \mid X) = \frac{p(X \mid Y)p(Y)}{p(X)} \propto p(X \mid Y)p(Y)$$

$$c(x) = \underset{Y \in \{pos, neg\}}{\arg\max}\ p(x \mid Y)p(Y)$$

Fit a model for p(X|Y) and for P(Y)

## Bayes classifier

Choose probability distribution M (e.g. MVN)

Fit $M_{pos}$ to all positive points: $p(X=x|pos) = M_{pos}(x)$

Fit $M_{neg}$ to all negative points: $p(X=x|neg) = M_{neg}(x)$

Estimate P(Y) from the class frequencies in the training data, or use domain-specific information.

$$c(x) = \underset{Y \in \{pos, neg\}}{\arg\max}\ p(x \mid Y)p(Y)$$

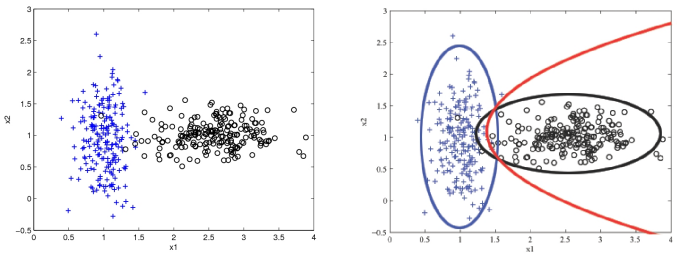
53

The generic Bayes classifier.

If we want a fully probabilistic classifier, we can normalise the value p(x|Y)p(Y) over the classes (essentially applying Bayes rule, and not using the simplification from the previous step).

---

## example for MVNs




54

Here is an example of what that looks like with 2 features. On the left we have two classes, blue and black. We fit a 2D normal distribution to each. Then, for a new point, we see which assigns the new point the highest probability density.

*source:* ***http://learning.cis.upenn.edu/cis520_fall2009/index.php?n=Lectures.NaiveBayes***

## Naive Bayes

Assume independence between all features, **conditional on the class**.

$$p(X_1, X_2 \mid Y) = p(X_1 \mid Y)p(X_2 \mid Y)$$

Often used with categoric features.



This works well for small numbers of features, but if we have many features, modelling the dependence between each pair of features gets very expensive. A crude, but very effective solution is Naive Bayes. NB just assumes that all features are independent, conditional on the class.

| "pill" | "meeting" | |
|---|---|---|
| T | T | spam |
| T | F | spam |
| T | T | ham |
| T | T | ham |
| F | T | ham |
| F | T | ham |
| F | T | ham |
| F | F | spam |
| T | F | spam |
| F | F | spam |
| F | F | ham |



Here is an example dataset, with binary features. Each feature indicates whether a particular word occurs in that instance.

We will build a naive Bayes classifier for this data by simply fitting a bernoulli distribution to each feature. That is, we will estimate p("pill"|spam) as the relative frequency with which the "pill" feature was true for spam emails.

| $X_1$ | $X_2$ | |
|---|---|---|
| T | T | spam |
| T | F | spam |
| **T** | T | **ham** |
| **T** | T | **ham** |
| **F** | T | **ham** |
| **F** | T | **ham** |
| **F** | T | **ham** |
| F | F | spam |
| T | F | spam |
| F | F | spam |
| **F** | F | **ham** |

$p(X_1=\mathbf{T} \mid \mathbf{ham}) = 2/6$

$p(X_1=\mathbf{F} \mid \mathbf{ham}) = 4/6$

57

Here is what Naive Bayes does: it selects all emails of one class, and then estimates the probability that X1 will be T as the relative frequency of emails for which X1 was T in the training set.

Strictly speaking, we are modelling $X_1$ as a Bernoulli distribution whose parameter we estimate as 2/6

---

| $X_1$ | $X_2$ | |
|---|---|---|
| **T** | T | **spam** |
| **T** | F | **spam** |
| T | T | ham |
| T | T | ham |
| F | T | ham |
| F | T | ham |
| F | T | ham |
| **F** | F | **spam** |
| **T** | F | **spam** |
| **F** | F | **spam** |
| F | F | ham |

$p(X_1=\mathbf{T} \mid \mathbf{spam}) = 3/5$

$p(X_1=\mathbf{F} \mid \mathbf{spam}) = 2/5$

58

We do the same for the spam class and for the other feature.

$$p(Y \mid X_1, \ldots, X_n) \propto p(X_1, \ldots, X_n \mid Y)p(Y)$$
$$= p(X_1 \mid Y) \times \ldots \times p(X_n \mid Y)p(Y)$$



This is the naive Bayes assumption formulaically. We simply factor p(X1,...Xn) into n separate, independent probabilities.

## smoothing

| $X_1$ | $X_2$ | |
|---|---|---|
| T | T | spam |
| T | F | spam |
| T | T | ham |
| T | T | ham |
| F | T | ham |
| F | T | ham |
| F | T | ham |
| T | F | spam |
| T | F | spam |
| T | F | spam |
| F | F | ham |

$p(X_1=\mathbf{T} \mid \mathbf{spam}) = 5/5$

$p(X_1=\mathbf{F} \mid \mathbf{spam}) = 0/5$



While Naive Bayes can work surprisingly well (given how strong and incorrect the assumption is), we do run into a problem if for some feature a particular value does not occur. In that case, we estimate the probability as 0.

$$
\begin{aligned}
p(Y \mid X_1, \ldots, X_n) &\propto p(X_1, \ldots, X_n \mid Y) p(Y) \\
&= p(X_1 \mid Y) \times \ldots \times p(X_n \mid Y) p(Y) \\
&= 0 \times \ldots \times p(X_n \mid Y) p(Y) \\
&= 0
\end{aligned}
$$



Since the whole estimate of our probability is just a long product, if one of the factors becomes zero, the whole things collapses. Even if all the other features gave this class a very high probability, that information is lost.

## pseudo-observations (aka Laplace smoothing)

| $X_1$ | $X_2$ | |
|---|---|---|
| T | T | spam |
| T | F | spam |
| T | T | ham |
| T | T | ham |
| F | T | ham |
| F | T | ham |
| F | T | ham |
| T | F | spam |
| T | F | spam |
| T | F | spam |
| F | F | ham |
| **F** | **F** | **spam** |
| **T** | **T** | **spam** |



To remedy this, we need to apply smoothing. The simplest was to do that is to add pseudo-observations. For each possible value, we add an instance where all the features have that value.

(We should do the same for the class ham).

unsmoothed

$$p(X_1 = T \mid Y = \text{spam}) = \frac{\text{freq. of T in spam data}}{\text{total \# of spam instances}}$$

smoothed

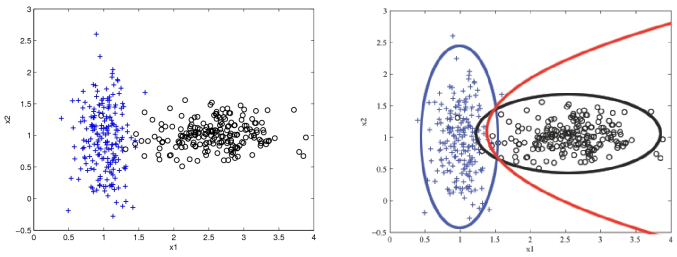
$$p(X_1 = T \mid Y = \text{spam}) = \frac{\text{freq. of T in spam}}{\text{total \# of spam instances} + v}$$

63

This changes our estimates as shown here (i.e. we don't actually need to add the pseudo-observations, we just change our estimator).

Here, v is the number of different values $X_1$ can take.

## continuous Naive Bayes



64

Naive Bayes is commonly associated with categoric features (to which bernoulli or categoric distributions are fitted), but it can also be used with numeric features.

In fact, in this example, fitting a univariate normal distribution to each feature independently would give exactly the same result (since the features are not correlated).

*source:* ***http://learning.cis.upenn.edu/cis520_fall2009/index.php?n=Lectures.NaiveBayes***

## summary so far

Bayesian vs frequentist learning. Use what works, mix-and-match.

Discriminative modeling: learn p(Y|X) directly

Generative modeling: learn p(X|Y) and p(Y)
Bayesian classifier, Naive Bayesian classifier

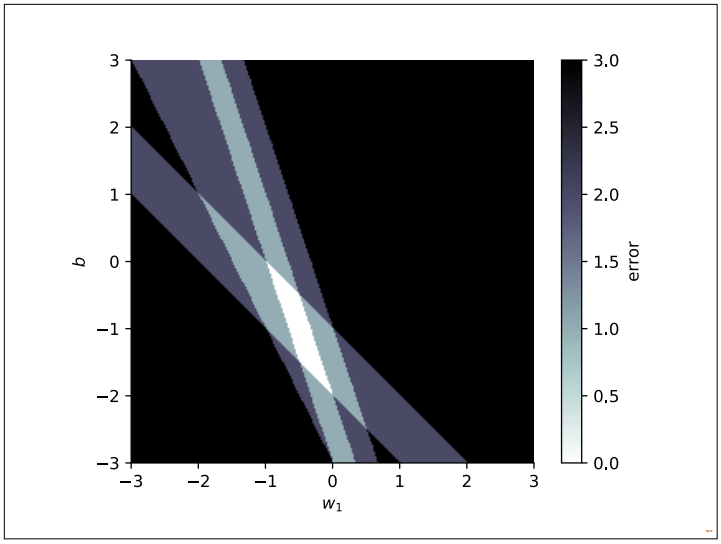
Naive Bayes: assumes independent features (conditional on the class).

Laplace smoothing: add pseudo-observations to avoid zero probabilities.



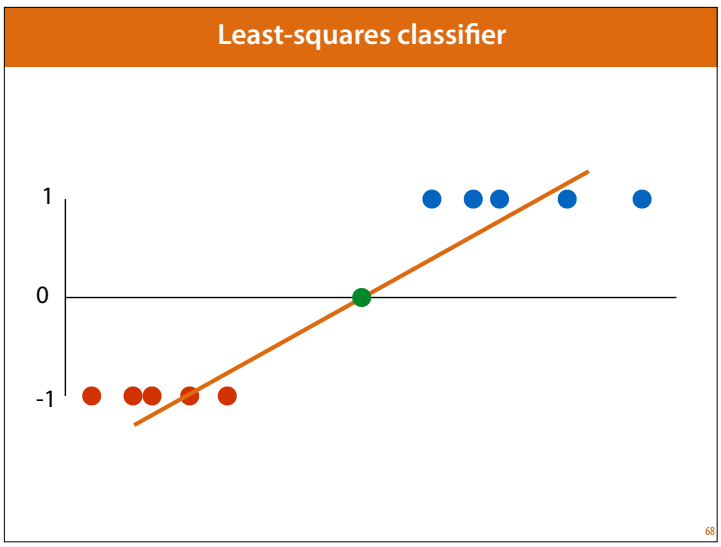
## discriminative classifier

Learn P(Y | X) directly.



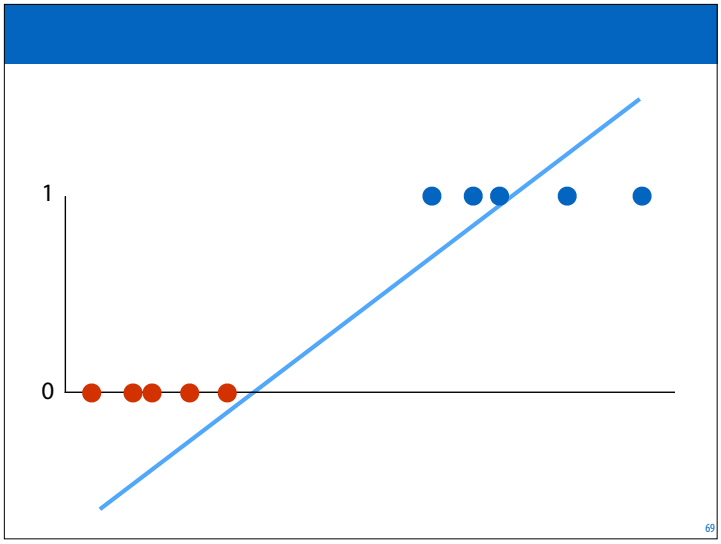

Remember that we were still on the lookout for good loss functions for the classification problem. We'll use the language of probability (and information theory) to define one for us.

---




This was our last attempt: the least squares loss.

Our thinking was: the hyperplane classifier check if **w**$\mathbf{x}$ + b is positive or negative, to decide whether to assign classes blue or red, respectively. Why not just give blue and red some arbitrary positive and negative values, and treat it as a regression problem.

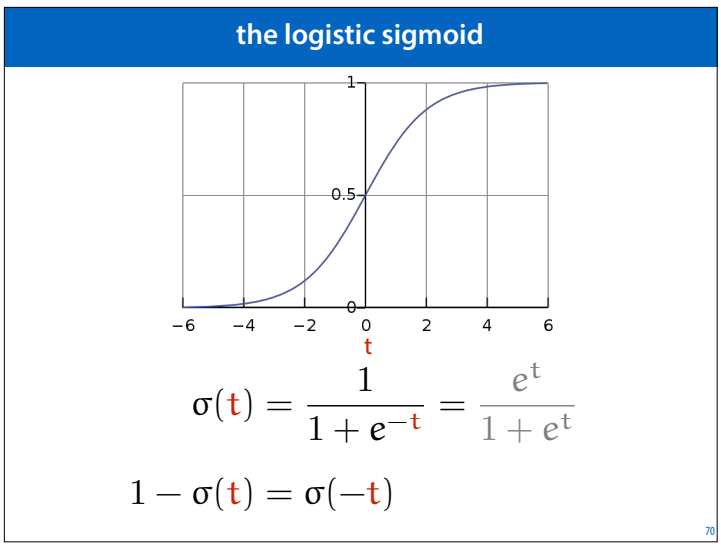



Here is another option: instead of given red and blue arbitrary values, we given them probabilities: the probability of being blue, which is 1 for all blue points and 0 for all red points. (The red points have moved from -1 to 0).

This doesn't look substantially different to our linear classifier because our function $\mathbf{w}^T\mathbf{x} + b$ still ranges from negative infinity to positive infinity. It doesn't produce probabilities, except over a very narrow range.

What we need, is a way to squeeze that whole range into the range [0, 1], so that the model produces probability values over the whole domain.

---

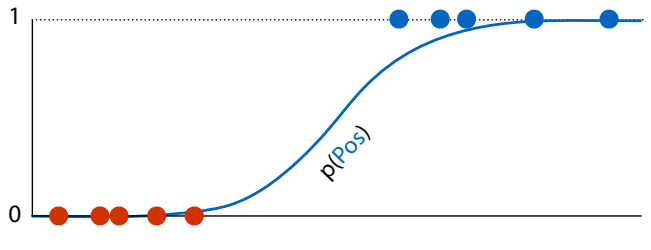



Enter the logistic sigmoid. Note that its domain is the entire real number line, and the range is [0,1]

An interesting property of the logistic sigmoid is the symmetry given in the second line. Basically the remainder between sigma(t) and 1, is itself a sigmoid running in the other direction.

*source: By Qef (talk) - Created from scratch with gnuplot, Public Domain, https://commons.wikimedia.org/w/index.php?curid=4310325*

$$c(x) = \sigma(\mathbf{w} \cdot \mathbf{x} + b)$$




71

This is our new classifier: we compute the linear function as before, but we apply the logistic sigmoid to the result, squeezing it into the interval [0, 1]. This means that we can interpret the output as the probability of the positive class.

Now all we need is a loss function that treats our outputs and our target data as probability distributions.

## cross-entropy loss

x: some data point

$q_x$: our classifier $q_x(C) = q(C|x)$

$q_x(\text{Pos}) = 0.1 \quad q_x(\text{Neg}) = 0.9$

$p_x$: the data label (Pos)

$p_x(\text{Pos}) = 1 \quad p_x(\text{Neg}) = 0$

72

This is where we come back to the cross entropy. We saw earlier that the cross entropy was a good measure of how different two distributions are (and the lower the cross entropy, the closer the two distributions).

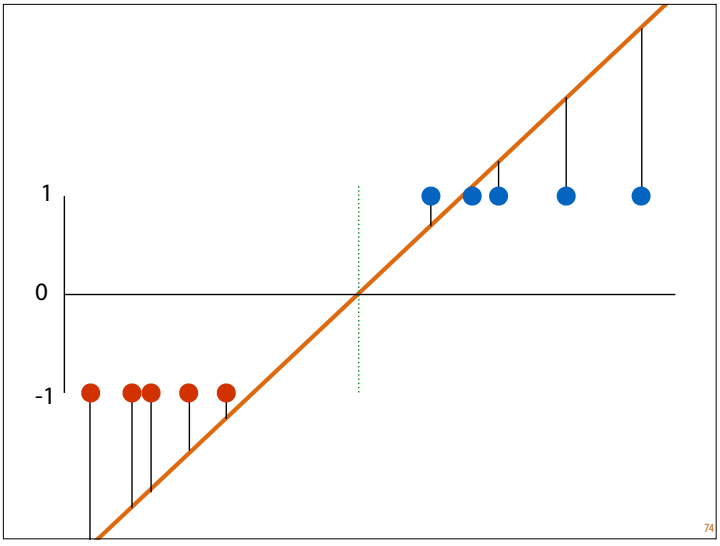
This makes the cross entropy a great loss function for models that output probabilities.

All we need to do is interpret the labeled data as a distribution too; a distribution that indicates that we are certain that the given label is true.

## cross-entropy loss

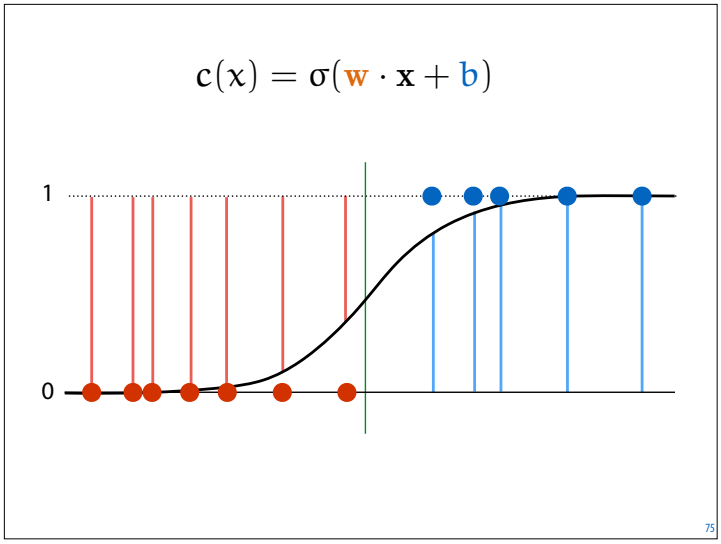
$$\begin{aligned}\text{loss}(q) &= \sum_{x \in X} H(p_x, q_x) \\ &= -\sum_{x \in X} (p_x(P) \log q_x(P) + p_x(N) \log q_x(N)) \\ &= -\sum_{x \in X_P} \log q_x(P) - \sum_{x \in X_N} \log q_x(N)\end{aligned}$$

73



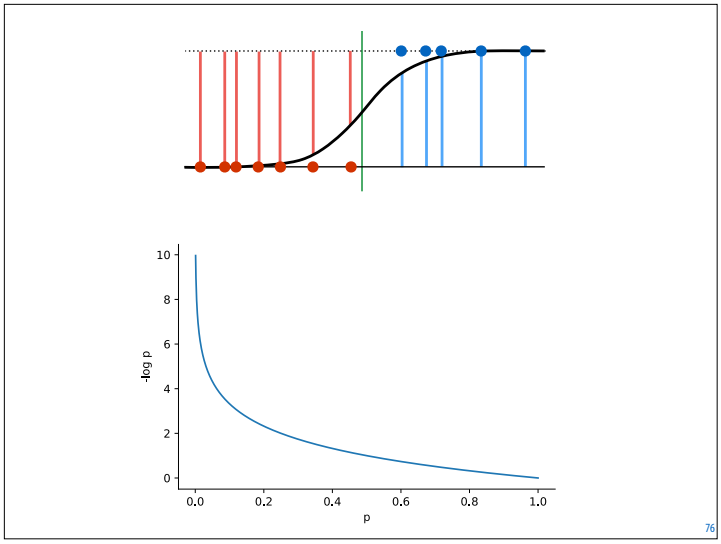

74

In the least-squares case, the loss function could be thought of in terms of the residuals between the prediction and the true values.




For the cross entropy loss, we can imagine the residuals for logistic regression as the lines drawn here. The cross entropy loss tries to maximise these lines by minimising the negative of their logarithm.

---




Remember that in the least squares loss we squared the residuals before summing them, to punish outliers. Taking the logarithm has a similar effect. Low probabilities are disproportionately punished.

## working out the gradient

$$\frac{\partial \text{loss}(\mathbf{w}, b)}{\partial w_i} = \frac{\partial \left(-\sum_{x \in X_P} \log q_x(P) - \sum_{x \in X_N} \log q_x(N)\right)}{\partial w_i}$$
$$= -\sum_{x \in X_P} \frac{\partial \log q_x(P)}{\partial w_i} - \sum_{x \in X_N} \frac{\partial \log q_x(N)}{\partial w_i}$$



We'll show you the basics of working out the gradient for logistic regression. The loss breaks apart in separate terms for the positive and negative points. Let's look at one of the positive terms (the negative can be derived in a similar way).

---

$$\frac{\partial \log q_x(P)}{\partial w_i} = \frac{\partial \log \sigma(\mathbf{w} \cdot \mathbf{x} + b)}{\partial w_i}$$
$$= \frac{\partial \log \frac{1}{1+\exp(-\mathbf{w}^T\mathbf{x}-b)}}{\partial w_i} = -\frac{\partial \log(1+\exp(-\mathbf{w}^T\mathbf{x}-b))}{\partial w_i}$$
$$= -\frac{\partial \log(1+\exp(-\mathbf{w}^T\mathbf{x}-b))}{\partial(1+\exp(-\mathbf{w}^T\mathbf{x}-b))} \frac{\partial(1+\exp(-\mathbf{w}^T\mathbf{x}-b))}{\partial w_i}$$
$$= -\frac{1}{\ln 2} \frac{1}{(1+\exp(-\mathbf{w}^T\mathbf{x}-b))} \frac{\partial \exp(-\mathbf{w}^T\mathbf{x}-b)}{\partial w_i}$$
$$= -\frac{1}{(1+\exp(-\mathbf{w}^T\mathbf{x}-b))} \frac{\partial \exp(-\mathbf{w}^T\mathbf{x}-b)}{\partial(-\mathbf{w}^T\mathbf{x}-b)} \frac{\partial(-\mathbf{w}^T\mathbf{x}-b)}{\partial w_i}$$
$$= -\frac{\exp(-\mathbf{w}^T\mathbf{x}-b)}{(1+\exp(-\mathbf{w}^T\mathbf{x}-b))} \cdot -x_i = (1-\sigma(\mathbf{w}^T\mathbf{x}+b))x_i$$
$$= q_x(N)x_i$$



Let's work out

Note that despite the intimidating formulas in the middle, the result is actually very simple. This is one of the properties of the logistic sigmoid, it tends to cancel itself out when the derivative is taken.

We ignore the constant multiplier (1/ln2) in the fourth line, because it doesn't change the direction of the gradient, only the magnitude. When we apply gradient descent we scale the gradient by a constant multiplier anyway, so we can ignore it. (Another option is to use the natural logarithm in the definition of the cross entropy).

d $\log_b(x)$/ dx = (1 /(ln b)) (1/x)

$$\frac{\partial \log q_x(P)}{\partial w_i} = \frac{\partial \log \sigma(w \cdot x + b)}{\partial w_i}$$
$$= \frac{\partial \log \frac{1}{1+\exp(-w^T x - b)}}{\partial w_i} = -\frac{\partial \log(1+\exp(-w^T x - b))}{\partial w_i}$$
$$= -\frac{\partial \log(1+\exp(-w^T x - b))}{\partial(1+\exp(-w^T x - b))} \frac{\partial(1+\exp(-w^T x - b))}{\partial w_i}$$
$$= -\frac{1}{\ln 2}\frac{1}{(1+\exp(-w^T x - b))} \frac{\partial \exp(-w^T x - b)}{\partial w_i}$$
$$= -\frac{1}{(1+\exp(-w^T x - b))} \frac{\partial \exp(-w^T x - b)}{\partial(-w^T x - b)} \frac{\partial(-w^T x - b)}{\partial w_i}$$
$$= -\frac{\exp(-w^T x - b)}{(1+\exp(-w^T x - b))} \cdot -x_i = (1 - \sigma(w^T x + b))x_i$$
$$= q_x(N)x_i$$

79

Let's work out

Note that despite the intimidating formulas in the middle, the result is actually very simple. This is one of the properties of the logistic sigmoid, it tends to cancel itself out when the derivative is taken.

We ignore the constant multiplier (1/ln2) in the fourth line, because it doesn't change the direction of the gradient, only the magnitude. When we apply gradient descent we scale the gradient by a constant multiplier anyway, so we can ignore it. (Another option is to use the natural logarithm in the definition of the cross entropy).

d $\log_b(x)$/ dx = (1 /(ln b)) (1/x)

---

$$\frac{\partial \text{loss}(w, b)}{\partial w_i} = -\sum_{x \in X_P} q_x(N)x_i + \sum_{x \in X_N} q_x(P)x_i$$

80

## logistic regression

Use the sigmoid function to turn a linear classifier into a discriminative probabilistic *classifier*.

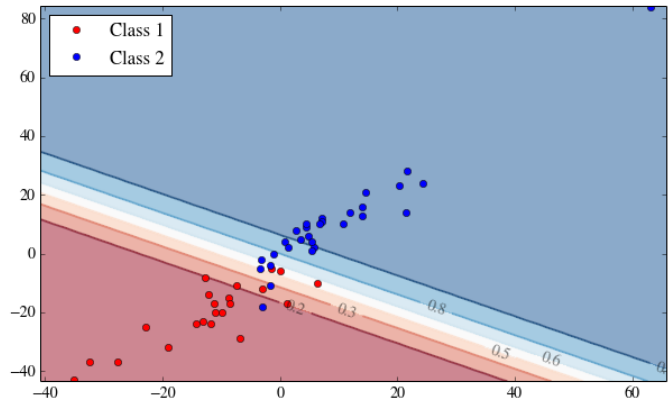
Use cross entropy loss.

Derive the gradient and search for good weights.

No analytical solution, but the problem is convex.

81

*Regression* is a bit of misnomer, since we're building a classifier. I suppose the confusing terminology comes from the fact that we're fitting a line through the probability values in the data.

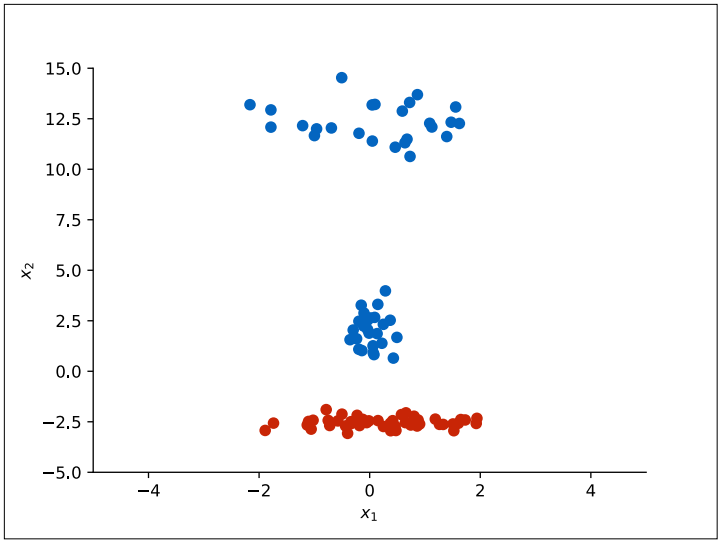



82

Here's what it looks like if we get it right. The areas far from the decision boundary have high red and blue probabilities. The closer we get, the more even the probabilities become and on the decision boundary, the probability is exactly 50/50.

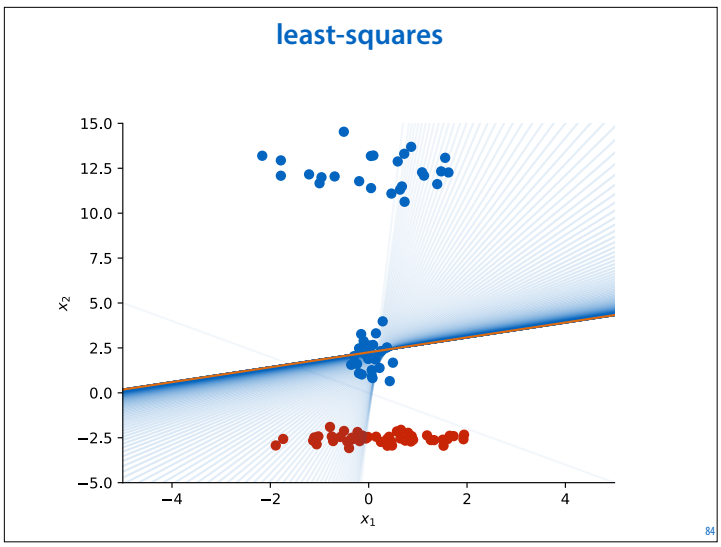
And remember, we can add cross products to make it non-linear.

source: **https://stackoverflow.com/questions/20045994/how-do-i-plot-the-decision-boundary-of-a-regression-using-matplotlib**



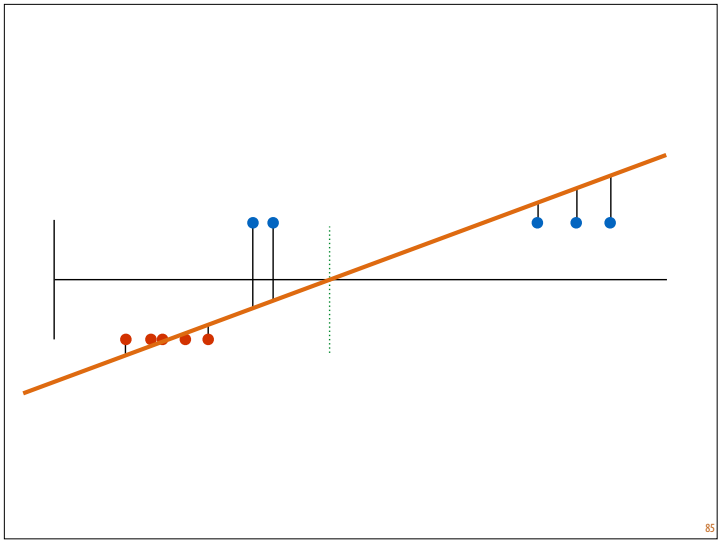

Here is a 2D dataset that shows a common failure case for the least square classifier. The points at the top are so far away from the ideal decision boundary that they would have huge residuals under the least squares model.

---

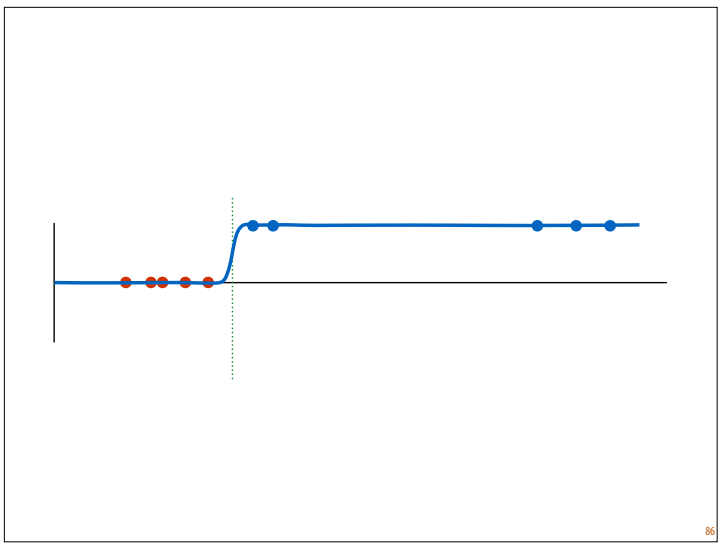



Here is what the least-square regression converges to. Clearly, this is not a satisfying solution for such an easily separable dataset. The blue points at the top are so far from the decision boundary.

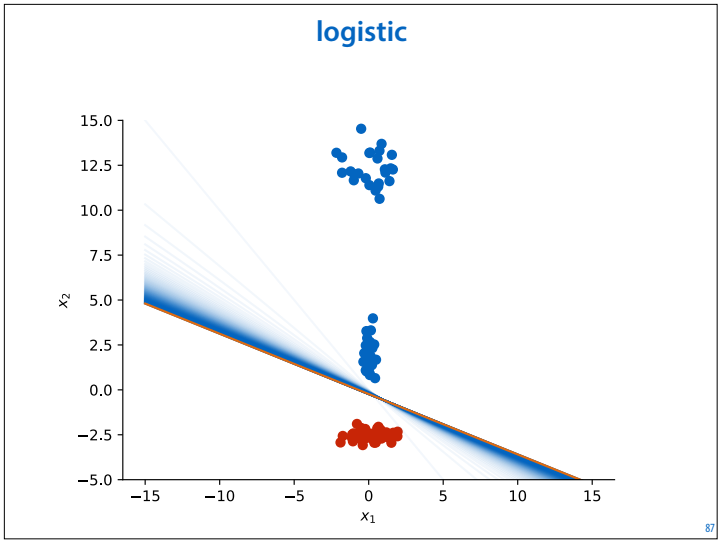
In the linear models 1 lecture, we fixed one of the parameters to 1, so that we could plot the loss surface. This time, we're optimizing all three parameters.




Here is a 1D view of a similar situation. To minimise the residuals of the blue outliers, the classifier is forced to move the decision boundary to the right.

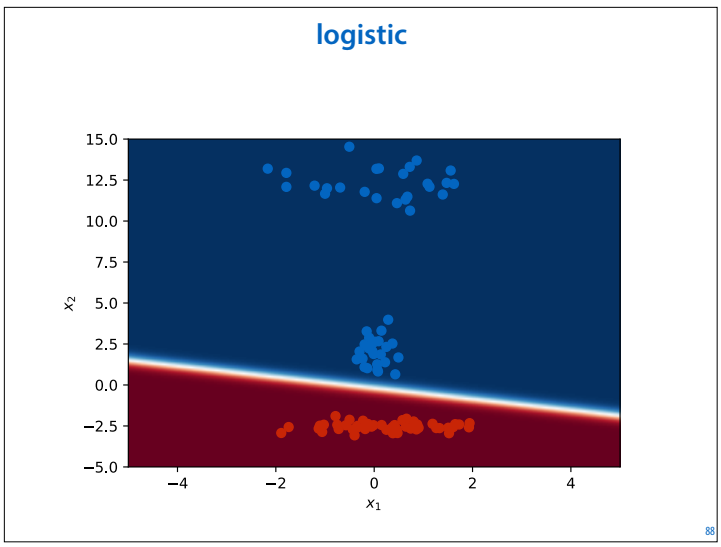



The logistic model doesn't have this problem.If the model fits well around the decision boundary, it doesn't have to worry at all about points that are far away (if they're on the right side of the boundary),



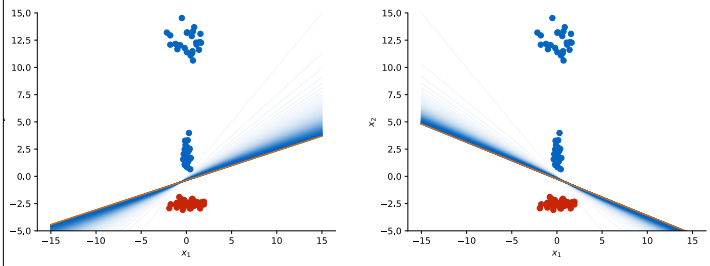

And here is the logistic regression classifier.

---




And here is the probability function (blue is high probability of positive, red is high probability of negative).

## logistic




next lecture: maximum margin classifier

89

Note that for such well-separable classes, there are many suitable classifier, and logistic regression has no reason to prefer one over the other (all points are assigned the correct probability very close to 1). We'll see a solution to this problem next lecture, when we meet our final loss function: the SVM loss.

---

## classification losses

~~Least squares loss (today)~~

~~Log loss / logistic regression (week 3, Probability 1)~~

SVM loss (week 3, Linear Models 2)

90

## summary: logistic regression

Use logistic sigmoid to provide class probabilities from a linear classifier

Use **cross-entropy as a loss function**

Points near the decision boundary get more influence than points far away.
The opposite is true for the least squares classifier.

Cross-entropy loss generalises naturally to multiclass classification (more next week).



## summary

**probability**

probability theory
mathematical framework for describing probability

statistics
applying probability theory to the real world

objective probability
e.g. Bayesianism

subjective probability
e.g. frequentism

information theory
relates probabilities to descriptions (codes)
entropy, cross-entropy, KL-divergence

probabilistic classification

generative classifiers
model p(X|Class), apply Bayes to get p(Class|X)

optimal Bayes classifier
full Bayesian treatment

Bayes classifier
point estimate for P(X|Cls)

naive Bayes classifier
Assume class-conditional independence

discriminative classifiers
model p(Class|X) directly

logistic regression
linear model + sigmoid with cross-entropy loss

mlcourse@peterbloem.nl